# Homework 2 Report

Tiantu Xu

## API (Class: `NeuralNetwork`)

```
-- create the dictionary of matrices Θ
[nil] __init__(([int] in, [int] h1, [int] h2, …, [int] out))
-- returns Θ(layer)
[2D DoubleTensor] getLayer([int] layer)
-- feedforward pass transposed design matrix
[ND DoubleTensor] forward([ND DoubleTensor] input)
```

## Secondary API (`logic_gates`)

```
Class: [boolean] AND([boolean] x, [boolean] y)
Class: [boolean]  OR([boolean] x, [boolean] y)
Class: [boolean] NOT([boolean] x)
Class: [boolean] XOR([boolean] x, [boolean] y)
```

### Major work:

### Part A:

(1) In class NeuralNetwork, the argument is stored as a list, which contains size for 1 input layer, multiple hidden layers, and 1 output layer.
(2) The theta is initialized as a dictionary, with a mean=0 and std=1/sqrt(layer_size) using torch.normal()
(3) getLayer() will return the theta of a specific layer
(4) forward() will perform matrix multiplication with torch.mm(). Before each calculation, the input will add a bias dimension using torch.cat(), and the theta is transposed using torch.t()

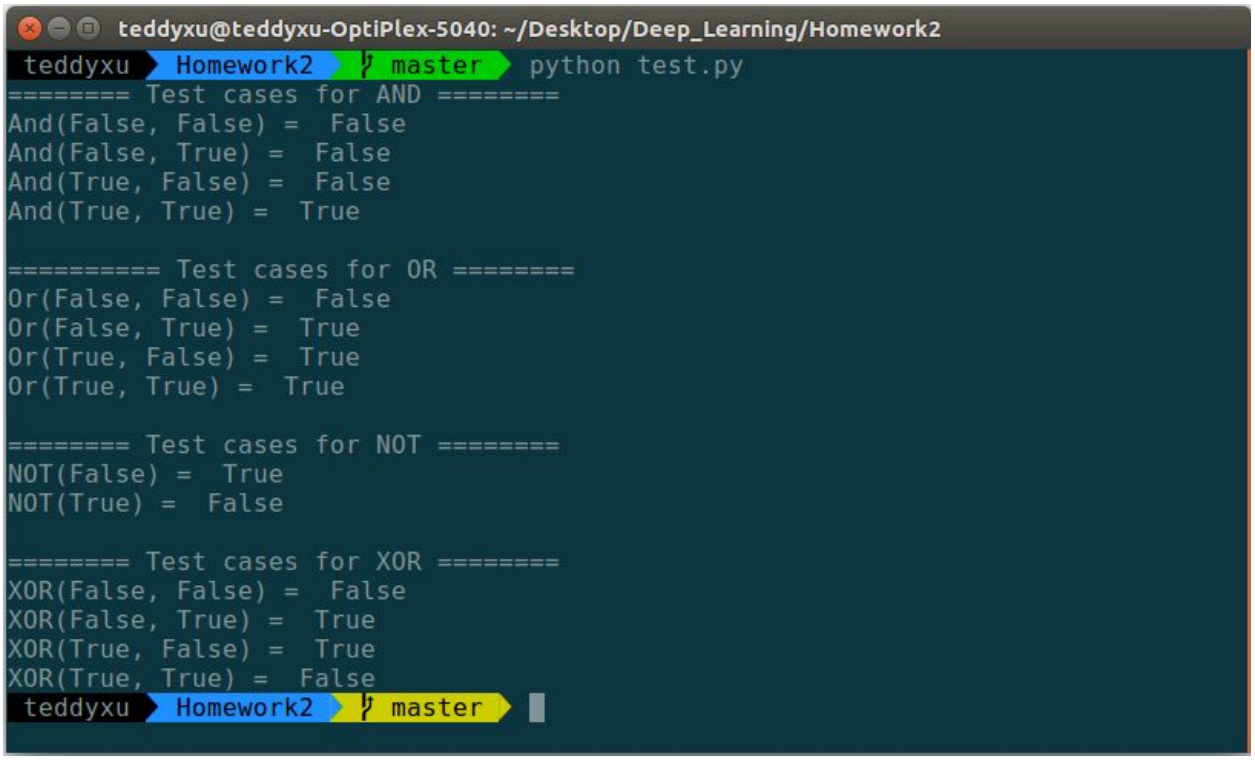
### Part B:

getLayer() will return layers. The weights of AND, OR, NOT and XOR gates are defined as
AND: ([-3], [2], [2])
OR: ([-2], [3], [3])
NOT: ([0], [-1])
XOR: ([-2, -2], [3, -3], [-3, 3]), ([-2], [3], [3])

### How to run:

$ python test.py

## Result:




## Conclusion:

The logic gates successfully perform the logic operations through a feed forward neural network.